$mathbf{QC^2AS}$: 電子相関理論のための数式処理システムの設計とその利用 (Computer Algebra : Design of Algorithms, Implementations and Applications)

AUTHOR(S):

小副川, 健; 望月, 祐志; 横山, 和弘

数理解析研究所講究録
第1815巻 2012年 152-162

QC^{2}AS—電子相関理論のための数式処理システムの設計とその利用

QC^{2}AS—Design of a computer algebra system for electron correlation theory and its application

小副川 健
TAKESHI OSOEKAWA
立教大学
RIKKYO UNIVERSITY *

望月 祐志
YUJI MOCHIZUKI
立教大学
RIKKYO UNIVERSITY †

横山 和弘
KAZUHIRO YOKOYAMA
立教大学
RIKKYO UNIVERSITY ‡

Abstract

本稿では, 量子化学の記号計算のための数式処理システム QC2AS を紹介する. QC2AS は, 計算機代数の非専門家が, 計算機代数的手法を利用できるようにする手段として設計されている. 本稿ではこのシステムの設計を, いくつかの計算例と共に述べる.

Abstract

In this paper, we introduce QC2AS, a computer algebra system for symbolic quantum chemical computations. QC2AS has been designed to provide an access to our computer algebraic techniques without knowing the background theories. In this paper, we illustrate its design and some examples.

1 はじめに

本稿は, 電子相関理論に現れる代数表式を扱う数式処理システムとして著者らが設計・開発している **QC^{2}AS** を紹介するものである.

電子相関理論は量子化学において, 高精度の数値計算をするうえで必要な理論で, 電子相関理論を用いた計算には**テンソル縮約式**と呼ばれる**代数表式**が頻出する [17, 16]. これらの式を量子化学の数値計算に利用するには, この代数表式を**記号計算**で簡約するなどし, 数値計算できる形に変形する必要がある. 計算結果の信頼性を高めるためには, より高次の代数表式を用いることが必要だが, 代数表式の次数が上がるにつれて式の複雑さは飛躍的に増すため, 人手による代数表式の操作ではすぐに限界がくる.

本研究では, 電子相関理論に現れる代数表式の操作を自動化するアルゴリズムを構築し, それを実装して, 量子化学の研究者自身が使えるような手段を提供することを目標にしている. 計算機代数の研究としては, 応用分野の開拓という意義の他に, 今まで注目してこなかったような問題に焦点を当てることで, 新たな計算手法の創発の可能性がある.

本研究で構築したアルゴリズムを利用できる手段は, 計算機代数の知識を必ずしも持っていなくても使えることが望ましい. また, 主用途の計算は実行効率が良いように, 低レベルのプログラム言語によって書か

*osoken@rikkyo.ac.jp
†fullmoon@rikkyo.ac.jp
‡yokoyama@rkmath.rikkyo.ac.jp

れた実装が理想である. これらのことから, 既存の数式処理システムのパッケージとして実装するのは, 十分とは言えない. 一方で, 様々な代数表式の扱いに対応するために, 拡張性は高い実装であることが望ましい. 以上のことから, 我々は電子相関理論の代数表式の扱いに特化した数式処理システムの開発への着想を得た [18]. この数式処理システムを Quantum Chemistry と Computer Algebra System の頭文字を取って QC²AS と名付け, 著者らが独自に設計・開発を行っている[1]. 最新版のバイナリは QC²AS のウェブサイト (`http://www2.rikkyo.ac.jp/web/fullmoon/qc2as/`) で入手できる.

以下の式は代表的な電子相関理論である Møller-Plesset 摂動論における 2 次の代数表式である.

$$E_{\mathrm{MP2}} = \frac{1}{4}\left(\sum_{ijab} \frac{[ia|jb][ai|bj] - [ia|jb][aj|bi] - [ib|ja][ai|bj] + [ib|ja][aj|bi]}{\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b}\right). \tag{1}$$

式における i, j, a, b は分子軌道のインデックス, ε_i など軌道 i の**軌道エネルギー**を意味する実数で, また $[ij|ab]$ は **2 電子積分**であり

$$[rs|tu] = \int \chi_r^*(\mathbf{x}_1)\chi_s(\mathbf{x}_1) r_{12}^{-1} \chi_t^*(\mathbf{x}_2)\chi_u(\mathbf{x}_2) d\mathbf{x}_1 d\mathbf{x}_2, \tag{2}$$

である. これらの物理的な意味については [17] を参照されたい. 問題は, $[rs|tu], \epsilon_{ijab} := (\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b)^{-1}$ を変数として, 代数表式をこれらの多項式と見做した時の式操作の自動化である. これらの変数は定義から

$$[rs|tu] = [sr|tu] = [rs|ut] = [sr|ut] = [tu|rs] = [ut|rs] = [tu|sr] = [ut|sr], \tag{3}$$

$$\epsilon_{ijab} = \epsilon_{jiab} = \epsilon_{ijba} = \epsilon_{jiba}, \tag{4}$$

を満たし, また, 分子軌道のインデックス i と j, a と b はそれぞれ等価であり, 入れ替えられることに注意すると, 式 (1) は最終的に

$$E_{\mathrm{MP2}} = \frac{1}{2}\sum_{ijab} \frac{[ia|jb]^2 - [ia|jb][ib|ja]}{\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b} \tag{5}$$

という表現と同じであることがわかる.

これに対して**スピン積分**[17] を行い, 数値計算に使う式として

$$\sum_{ijab} \frac{2(ia|jb)^2 - (ia|jb)(ib|ja)}{\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b}$$

を得る. 数値計算は, 例えば

$$\begin{aligned}
(11|11) &= 4.744 \\
(21|11) &= -0.4177 \\
&\vdots \\
(71|11) &= -0.2379 \\
(12|11) &= -0.4177 \\
&\vdots \\
(77|77) &= 0.5932 \\
\varepsilon_1 &= -20.23 \\
&\vdots \\
\varepsilon_7 &= 0.5899
\end{aligned}$$

[1] 発音は [kju:kæs], キューキャス, キューカス.

といったテーブルが与えられ, 分子軌道のインデックスそれぞれに 1,... を当てはめたときの値を評価して $\sum$ の展開を行う. この例の場合, $i,j = 1,2,\ldots,5$ と $a,b = 6,7$ で展開を行うことになっており, 結果 -0.03586 を得る.

以上が量子化学の典型的な計算の流れであるが, このうちスピン積分までの部分は純粋に記号計算であり, 計算機代数の問題としてモデル化できる. 本稿に先立って著者らは [14, 18] で代数表式の簡約のアルゴリズムの構築とその実装を完成させている. そこで本稿ではこの実装を, 数式処理システム QC²AS の機能の一部として利用できるようにし, それを用いて実用レベルの問題である E_{MP4} の式の一部

$$
\begin{aligned}
E_{\mathrm{MP4}(S)} = \frac{1}{8} \sum_{\substack{ijklm \\ abcde}} a(ijab) \times \quad \Big(\quad & \langle ck||ed\rangle \big(a(iked)\langle ab||cj\rangle \epsilon_{ij} + a(jked)\langle ab||ic\rangle \epsilon_{jc} \big) \\
+ \quad & \langle kc||ml\rangle \big(a(mlac)\langle kb||ij\rangle \epsilon_{ka} + a(mlbc)\langle ka||ji\rangle \epsilon_{kb} \big) \\
- \quad & a(lkcd) \big(\langle ab||cj\rangle \langle id||lk\rangle \epsilon_{ic} + \langle ab||ic\rangle \langle jd||lk\rangle \epsilon_{jc} \big) \\
- \quad & a(kldc) \big(\langle ka||ij\rangle \langle al|dc\rangle \epsilon_{ka} + \langle ka||ji\rangle \langle bl||dc\rangle \epsilon_{kb} \big) \Big)
\end{aligned}
$$

($\epsilon_{ia} := (\varepsilon_i - \varepsilon_a)^{-1}$, $\langle ij||ab\rangle := [ia|jb] - [ib|ja]$, $a(ijab) := \langle ij||ab\rangle \epsilon_{ijab}$) を簡約した結果を, その計算時間などとともに紹介する.

2 節では QC²AS の基本的な設計について述べる. システムの形式上の中核であるシステム制御クラスについて 2.1 節で, そして実際の計算で中心的役割を果たす Q オブジェクトクラスについて 2.2 節でそれぞれ述べる. QC²AS がどのような動作をするかを 3 節で見る. 3.1 節では QC²AS の一般的な数式処理システムとしての動作の様子を, 3.2 節では量子化学の記号の扱いや, 簡約アルゴリズムの計算結果などを紹介する. また, 簡約操作にかかった計算時間を 3.3 節で示す.

2 システムの設計

本節では QC²AS の基本的な構成と, QC²AS で扱われるオブジェクトの単位である Q オブジェクトについて述べる. QC²AS は現在およそ 14,000 行の C++のプログラムでできている.

QC²AS は単純な構成で十分な機能を実現できるように, 注意深く設計されている. 構成要素は**システム制御クラス**と**Q オブジェクトクラス**に大別できる. 基本的な設計思想は, システム制御クラスをなるべく小さくし, 各クラスの独立性を高いまま保っておくことである.

2.1 システム制御クラス

システム制御クラスは実際には, それぞれ役割の異なる複数のクラスで構成されている. システム制御クラスで管理しているのは, 以下のものである.

- 構文解析器
- スコープ (大域, 局所)
- ログ
- 環境変数

システム制御クラスは, ユーザからの入力の文字列を受け取ると, これを構文解析器で Q オブジェクトに変換し, Q オブジェクトを評価する命令を出す. そして Q オブジェクトの評価が終わると, 結果を文字列に変換し, コンソールに表示する.

各 Q オブジェクトの評価の方法や, コンソールに表示する形式は, Q オブジェクトクラス側のメソッドとして実装されている. これによって, システム制御クラスは, 一連の処理の中で今扱っている Q オブジェクトが具体的にどういうデータなのかを把握する必要がない. ユーザの入力が意味する式によって, 型の違う Q オブジェクトが生成されるし, 型によって処理する方法も異なるが, システム制御クラスは評価する命令を出し, 評価が終わったらコンソールに表示する命令を出すだけである. これはつまり, Q オブジェクトのデータ型を増やしたり, Q オブジェクトの内部表現を変更しても, システム制御クラスに一切変更を加える必要がないことを意味している. これはオブジェクト指向言語による開発のメリットと言える.

2.2 Q オブジェクトクラスの概要

QC²AS で扱われるデータは, ほとんど全て Q オブジェクトクラスのインスタンスである. Q オブジェクトクラスの QC²AS における役割は次の 2 つである.

- データの保持
- 命令に対する処理方法の実装

整数型などの具体的なデータの型は, Q オブジェクトクラスのクラス内クラスとして実装されている. 上記の 2 つの役割は, 実際はこのクラス内クラスが行っている. Q オブジェクトクラスのインスタンスに出された命令は, このクラス内クラスへの命令に変換され, 該当するメソッドが呼び出される.

Q オブジェクトに対する基本命令は次のようなものがある. 括弧内は, 明示的に QC²AS から呼び出すときの関数名である.

- 評価する (`Eval`)
- 型を取得する (`Type`)
- 大きさを取得する (`Size`)
- 画面に出力する (`Print`)
- 部分を取り出す (`Part`)

また, あるデータ型と特別な演算が定義されている場合は, そのデータ型を引数にとって演算を行う命令も定義されている.

Q オブジェクトのデータ型は, 具体的なデータだけでなく, 演算などの操作も Q オブジェクトのインスタンスとして実体が作られる. これは遅延評価を実現するためで, これらの実体はシステム制御クラスから評価する命令が出されるまで保持される. これと基本命令の組合せによって, QC²AS の全ての機能が実現されている.

以下, 主な Q オブジェクトのデータ型を紹介する. 括弧内は, QC²AS 上で `Type` 命令を出したときの返り値である.

- 整数 (`t_Integer`)
- 浮動小数 (`t_Float`)

- 有理数 (t_RationalNumber)
- 多項式 (t_Polynomial)
- 項 (t_Term)
- 識別子 (t_Identifier)
- リスト (t_List)
- 2 電子積分 (t_QCObj)
- 軌道の添字 (t_OrbIdx)
- エラー情報 (t_ErrorInfo)
- 型情報 (t_TypeInfo)

これ以外にも真偽値型や文字列型, 演算や関数呼出などの操作を意味する型など, 多くの型が実装されている. これらは基本的に独立しており, このことが保守を容易にしている.

3 計算例とタイミングデータ

この節では QC²AS を使った計算の例を紹介する. QC²AS はコンソール上で動作し, 多くの数式処理システムと同様に, ユーザの入力に対して対話的に出力を返す. 以下, ">" で始まる行はコンソールに対するユーザの入力を意味し, "=" で始まる行は QC²AS からの出力を意味する.

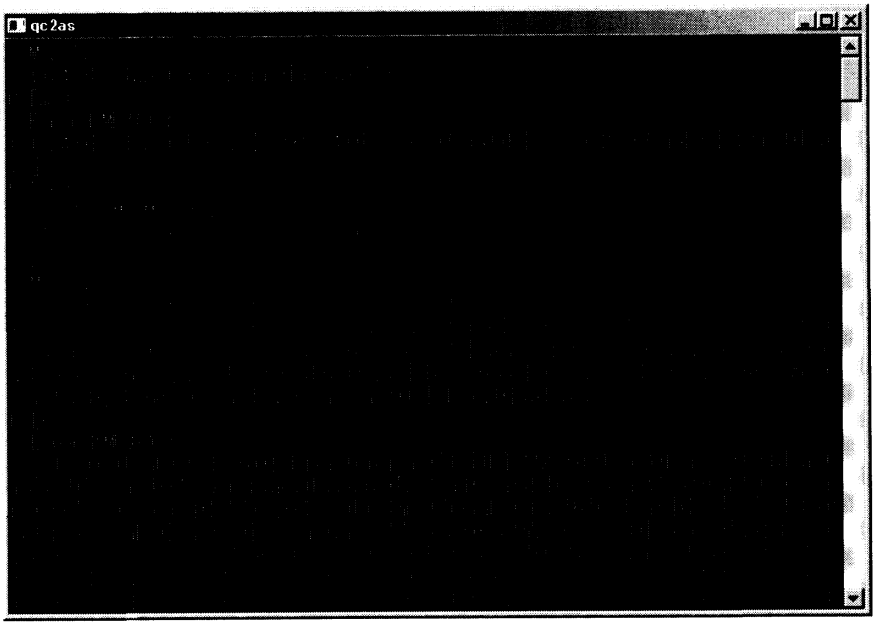

図 1: QC²AS のスクリーンショット

3.1 基本的な数式処理システムとしての動作

QC²AS は量子化学の記号計算がターゲットだが, 基本部分は (普通の) 多項式を扱う数式処理システムとして動作するように設計されている.

数を意味する Q オブジェクトには整数型, 浮動小数型, 有理数型などがある. 型をまたいだ演算もでき, 計算結果の型は自動的に適切なものに変換される. 例えば整数に整数を足すと整数型であり, 浮動小数型に整数を足すと浮動小数型になる.

```
> 1+1;
= 2
> 1.2+1;
= 2.2
```

有理数は有理数のまま扱う. 計算結果は常に既約分数 (あるいは整数) に約分される. 有理数同士の演算も, 有理数のまま行う.

```
> 2/4;
= 1/2
> 1/2+1/3;
= 5/6
```

有理数と浮動小数の演算では, 有理数は一時的に浮動小数に変換され, 計算結果は浮動小数になる.

```
> 1/2+1.2;
= 1.7
```

このように, Q オブジェクトは型を動的に変えながら計算を行っているが, 型の変換は QC²AS によって適切に行われるため, ユーザは特に目的がない限り, Q オブジェクトの型について意識する必要がない.

分子軌道のインデックスとして予約されている文字以外の文字は不定元として扱うことができる. 現在 QC²AS では英字大文字と, 分子軌道のインデックスとして予約されている文字以外の英字小文字, そしてアルファベットから始まる 2 文字以上の文字列がこれに該当する. 2 文字以上続けて打っても 1 つの不定元として認識するので, 不定元の積は*で区切る.

```
> xy;
= xy
> x*y;
= y*x
```

多項式は内部で定義されている順序によってソートされる.

```
> x+y-1;
= y+x-1
> 2*x*y-z;
= -z+2*y*x
```

不定元はプログラム変数としても利用できる. 値を代入した不定元は, 以後プログラム変数として扱われる. QC²AS ではユーザが明示しない限り多項式は展開しない. 展開する場合は Expand 関数を呼び出す.

```
> A;
= A
> A=(x+1)^2;
= (x+1)^2
> Expand(A);
= x^2+2*x+1
```

関数名に使う文字列の制限は, プログラム変数のそれと全く同じである. つまりプログラム変数として使える文字列は, 関数名にも用いることができる. また, QC²AS の関数識別子は, 関数名と引数の数によって別々に管理されているため, 関数のオーバーロードを許す.

3.2 量子化学の記号の扱い

QC²AS では, 軌道のインデックスとして使う文字は起動時に登録される. デフォルトでは i, j, ..., n は占有軌道のインデックス, a, b, ..., f は仮想軌道のインデックスである. これは後でユーザが変えることもできる. これらの文字は不定元やプログラム変数とは振舞が異なり, 従って多項式の変数として扱うことはできない.

```
> x*x;
= x^2
> i*i;
= ERROR: undefined operation (i*i).
```

2 電子積分 $[ij|ab]$ は, そのまま入力できる.

```
> [ij|ab];
= [ij|ab]
```

2 電子積分の定義からくる等値性 (3) は, 暗に使われる. 分子軌道のインデックスは, 占有軌道か仮想軌道かで比較 (占有軌道が小さい) され, 同じ種類であれば, アルファベットの順番の大小で比較するという順序がつけられている. そして 2 電子積分は, 最も左にある分子軌道のインデックスを優先する辞書式順序で順序付けされている. QC²AS では (3) を全て使って得られる等価な 2 電子積分のうち, 最も順序の小さい 2 電子積分に自動的に書き換える.

```
> [ba|ji];
= [ij|ab]
> [ij|ab]==[ji|ab];
= True
> [ij|ab]==[ia|jb];
= False
```

2 電子積分の QC²AS 上での扱いは, 不定元とほぼ同じ (プログラム変数としては利用できない) である. 従って, 2 電子積分の和や積が使えて, それらの結果は 2 電子積分を変数に持つ多項式となる.

```
> [ij|ab]+[ji|ab];
= 2*[ij|ab]
> [ij|ab]*[ij|ba];
= [ij|ab]^2
```

E_{MP2} など, よく使われる式は組込関数として登録されている. ϵ_{ijab} は E(ijab) と表現されている. これも QC²AS 上の扱いは 2 電子積分と全く同じである.

```
> MP2();
= (1/4)*E(ijab)*((-[ib|ja]+[ia|jb])^2)
> Expand(MP2());
= (1/4)*E(ijab)*([ib|ja]^2)+(1/4)*E(ijab)*([ia|jb]^2)-(1/2)*E(ijab)*[ib|ja]*[ia|jb]
```

[18] で提案したアルゴリズムの実装は QC²AS の組込関数 SimplifyMP で利用できる. 引数は任意の代数表式を取る. 項の順序は分子軌道のインデックスの順序に基づいた組込の順序が使われており, 全ての項がその順序の下, 最も小さくなるように書き換えられる.

```
> SimplifyMP(MP2());
= (1/2)*E(ijab)*([ib|ja]^2)-(1/2)*E(ijab)*[ib|ja]*[ia|jb]
```

E_{MP3}[17] の簡約も, E_{MP2} と全く同様にしてできる.

```
> A=MP3();
= (1/8)*E(klab)*E(ijab)*(-[kb|la]+[ka|lb])*(-[ib|ja]+[ia|jb])*(-[il|jk]+[ik|jl])+(1/8)
*E(ijcd)*E(ijab)*(-[ad|bc]+[ac|bd])*(-[id|jc]+[ic|jd])*(-[ib|ja]+[ia|jb])-(1/4)*E(ijab
)*(E(jkbc)*(-[jc|kb]+[jb|kc])*(-[ia|kc]+[ik|ac])+E(jkac)*([jc|ka]-[ja|kc])*(-[ib|kc]+[
ik|bc])+E(ikbc)*([ic|kb]-[ib|kc])*(-[ja|kc]+[jk|ac])+E(ikac)*(-[jb|kc]+[jk|bc])*(-[ic|
ka]+[ia|kc]))*(-[ib|ja]+[ia|jb])
> A=Expand(A);
= -(1/8)*E(klab)*E(ijab)*[kb|la]*[ib|ja]*[il|jk]+(1/8)*E(klab)*E(ijab)*[kb|la]*[ib|ja]
*[ik|jl]+(1/8)*E(klab)*E(ijab)*[kb|la]*[ia|jb]*[il|jk]-(1/8)*E(klab)*E(ijab)*[kb|la]*[
ia|jb]*[ik|jl]+(1/8)*E(klab)*E(ijab)*[ka|lb]*[ib|ja]*[il|jk]-(1/8)*E(klab)*E(ijab)*[ka
|lb]*[ib|ja]*[ik|jl]-(1/8)*E(klab)*E(ijab)*[ka|lb]*[ia|jb]*[il|jk]+(1/8)*E(klab)*E(ija
b)*[ka|lb]*[ia|jb]*[ik|jl]+(1/4)*E(jkbc)*E(ijab)*[jc|kb]*[ib|ja]*[ia|kc]-(1/4)*E(jkbc)
*E(ijab)*[jc|kb]*[ib|ja]*[ik|ac]-(1/4)*E(jkbc)*E(ijab)*[jc|kb]*[ia|kc]*[ia|jb]+(1/4)*E
(jkbc)*E(ijab)*[jc|kb]*[ia|jb]*[ik|ac]-(1/4)*E(jkbc)*E(ijab)*[jb|kc]*[ib|ja]*[ia|kc]+(
1/4)*E(jkbc)*E(ijab)*[jb|kc]*[ib|ja]*[ik|ac]+(1/4)*E(jkbc)*E(ijab)*[jb|kc]*[ia|kc]*[ia
|jb]-(1/4)*E(jkbc)*E(ijab)*[jb|kc]*[ia|jb]*[ik|ac]-(1/4)*E(jkac)*E(ijab)*[jc|ka]*[ib|k
c]*[ib|ja]+(1/4)*E(jkac)*E(ijab)*[jc|ka]*[ib|kc]*[ia|jb]+(1/4)*E(jkac)*E(ijab)*[jc|ka]
*[ib|ja]*[ik|bc]-(1/4)*E(jkac)*E(ijab)*[jc|ka]*[ia|jb]*[ik|bc]+(1/4)*E(jkac)*E(ijab)*[
ja|kc]*[ib|kc]*[ib|ja]-(1/4)*E(jkac)*E(ijab)*[ja|kc]*[ib|kc]*[ia|jb]-(1/4)*E(jkac)*E(i
jab)*[ja|kc]*[ib|ja]*[ik|bc]+(1/4)*E(jkac)*E(ijab)*[ja|kc]*[ia|jb]*[ik|bc]-(1/4)*E(ikb
c)*E(ijab)*[ja|kc]*[ic|kb]*[ib|ja]+(1/4)*E(ikbc)*E(ijab)*[ja|kc]*[ic|kb]*[ia|jb]+(1/4)
*E(ikbc)*E(ijab)*[ja|kc]*[ib|kc]*[ib|ja]-(1/4)*E(ikbc)*E(ijab)*[ja|kc]*[ib|kc]*[ia|jb]
+(1/4)*E(ikbc)*E(ijab)*[jk|ac]*[ic|kb]*[ib|ja]-(1/4)*E(ikbc)*E(ijab)*[jk|ac]*[ic|kb]*[
ia|jb]-(1/4)*E(ikbc)*E(ijab)*[jk|ac]*[ib|kc]*[ib|ja]+(1/4)*E(ikbc)*E(ijab)*[jk|ac]*[ib
|kc]*[ia|jb]+(1/4)*E(ikac)*E(ijab)*[jb|kc]*[ic|ka]*[ib|ja]-(1/4)*E(ikac)*E(ijab)*[jb|k
c]*[ic|ka]*[ia|jb]-(1/4)*E(ikac)*E(ijab)*[jb|kc]*[ib|ja]*[ia|kc]+(1/4)*E(ikac)*E(ijab)
*[jb|kc]*[ia|kc]*[ia|jb]-(1/4)*E(ikac)*E(ijab)*[jk|bc]*[ic|ka]*[ib|ja]+(1/4)*E(ikac)*E
(ijab)*[jk|bc]*[ic|ka]*[ia|jb]+(1/4)*E(ikac)*E(ijab)*[jk|bc]*[ib|ja]*[ia|kc]-(1/4)*E(i
kac)*E(ijab)*[jk|bc]*[ia|kc]*[ia|jb]-(1/8)*E(ijcd)*E(ijab)*[ad|bc]*[id|jc]*[ib|ja]+(1/
```

```
8)*E(ijcd)*E(ijab)*[ad|bc]*[id|jc]*[ia|jb]+(1/8)*E(ijcd)*E(ijab)*[ad|bc]*[ic|jd]*[ib|j
a]-(1/8)*E(ijcd)*E(ijab)*[ad|bc]*[ic|jd]*[ia|jb]+(1/8)*E(ijcd)*E(ijab)*[ac|bd]*[id|jc]
*[ib|ja]-(1/8)*E(ijcd)*E(ijab)*[ac|bd]*[id|jc]*[ia|jb]-(1/8)*E(ijcd)*E(ijab)*[ac|bd]*[
ic|jd]*[ib|ja]+(1/8)*E(ijcd)*E(ijab)*[ac|bd]*[ic|jd]*[ia|jb]
```

多項式の項数は Size 関数によって取得できる.

```
> Size(A);
= 48
```

簡約前の E_{MP3} を展開したときの項数は 48 であった. これを SimplifyMP 関数を用いて簡約すると

```
> B=SimplifyMP(MP3());
= -(1/2)*E(klab)*E(ijab)*[kb|la]*[ib|ja]*[il|jk]+(1/2)*E(klab)*E(ijab)*[kb|la]*[ib|ja]
*[ik|jl]+E(jkbc)*E(ikac)*[jc|kb]*[ic|ka]*[ia|jb]-E(jkbc)*E(ikac)*[jc|kb]*[ic|ka]*[ij|a
b]-2*E(jkbc)*E(ikac)*[jc|kb]*[ia|kc]*[ia|jb]+2*E(jkbc)*E(ikac)*[jc|kb]*[ia|kc]*[ij|ab]
+E(jkbc)*E(ikac)*[jb|kc]*[ia|kc]*[ia|jb]-E(jkbc)*E(ikac)*[jb|kc]*[ia|kc]*[ij|ab]-(1/2)
*E(ijcd)*E(ijab)*[ad|bc]*[id|jc]*[ib|ja]+(1/2)*E(ijcd)*E(ijab)*[ad|bc]*[id|jc]*[ia|jb]
> Size(B);
= 10
```

となり, 10 項まで減る.

$E_{\mathrm{MP4}(S)}$ は展開すると 128 項の式になるため, 展開された式の QC^{2}AS 上での表示は割愛するが, E_{MP2} や E_{MP3} と全く同じ手順によって簡約することができる.

```
> MP4S();
= (1/8)*E(ijab)*([km|lc]-[kl|mc])*(E(kb)*E(lmbc)*([lc|mb]-[lb|mc])*([ia|jk]-[ik|ja])+E
(ka)*E(lmac)*([lc|ma]-[la|mc])*(-[ib|jk]+[ik|jb]))*(-[ib|ja]+[ia|jb])+(1/8)*E(ijab)*(E
(jc)*E(jkde)*([je|kd]-[jd|ke])*(-[ib|ac]+[ia|bc])+E(ic)*E(ikde)*([jb|ac]-[ja|bc])*([ie
|kd]-[id|ke]))*(-[ke|cd]+[kd|ce])*(-[ib|ja]+[ia|jb])+(1/8)*E(ijab)*(E(klcd)*([kd|lc]-[
kc|ld])*(E(kb)*([ia|jk]-[ik|ja])*(-[ld|bc]+[lc|bd])+E(ka)*(-[ld|ac]+[lc|ad])*(-[ia|jk]
+[ik|ja]))+E(klcd)*([kd|lc]-[kc|ld])*(E(jc)*([jl|kd]-[jk|ld])*(-[ib|ac]+[ia|bc])+E(ic)
*([jb|ac]-[ja|bc])*([il|kd]-[ik|ld])))*(-[ib|ja]+[ia|jb])
> Size(A=Expand(MP4S()));
= 128
> B=SimplifyMP(A);
= E(mc)*E(klbc)*E(ijac)*[kc|lb]*[kb|lm]*[ic|ja]*[ia|jm]-2*E(mc)*E(klbc)*E(ijac)*[kc|lb
]*[kb|lm]*[ic|ja]*[im|ja]+E(mc)*E(klbc)*E(ijac)*[kc|lb]*[km|lb]*[ic|ja]*[im|ja]-(1/2)*
E(ld)*E(klbc)*E(ijad)*[kc|bd]*[kc|lb]*[id|ja]*[id|jl]-(3/2)*E(ld)*E(klbc)*E(ijad)*[kc|
bd]*[kc|lb]*[id|ja]*[ia|jl]+(1/2)*E(ld)*E(klbc)*E(ijad)*[kc|bd]*[kc|lb]*[id|ja]*[il|jd
]+(3/2)*E(ld)*E(klbc)*E(ijad)*[kc|bd]*[kc|lb]*[id|ja]*[il|ja]+(1/2)*E(ld)*E(klbc)*E(ij
ad)*[kc|bd]*[kb|lc]*[id|ja]*[id|jl]+(3/2)*E(ld)*E(klbc)*E(ijad)*[kc|bd]*[kb|lc]*[id|ja
]*[ia|jl]-(1/2)*E(ld)*E(klbc)*E(ijad)*[kc|bd]*[kb|lc]*[id|ja]*[il|jd]-(3/2)*E(ld)*E(kl
bc)*E(ijad)*[kc|bd]*[kb|lc]*[id|ja]*[il|ja]+E(ke)*E(jkcd)*E(ikab)*[jd|ce]*[jd|kc]*[ib|
ae]*[ib|ka]-2*E(ke)*E(jkcd)*E(ikab)*[jd|ce]*[jd|kc]*[ib|ae]*[ia|kb]+E(ke)*E(jkcd)*E(ik
ab)*[jd|ce]*[jc|kd]*[ib|ae]*[ia|kb]
```

```
> Size(B);
= 14
```

簡約結果は 128 項から 14 項まで減っている.

$$
\begin{aligned}
E_{\mathrm{MP4}(S)} &= \sum_{\substack{ijklm\\abcde}} \frac{1}{2}\epsilon_{ld}\epsilon_{klbc}\epsilon_{ijad}[kc|bd][id|ja] \\
&\times \Big([kc|lb]\big([il|jd]-[id|jl]+3[il|ja]-3[ia|jl]\big)+[kb|lc]\big([id|jl]-[il|jd]+3[ia|jl]-3[il|ja]\big)\Big) \\
&+ \epsilon_{mc}\epsilon_{klbc}\epsilon_{ijac}[kc|lb][ic|ja]\Big([kb|lm][ia|jm]-2[kb|lm][im|ja]+[km|lb][im|ja]\Big) \\
&+ \epsilon_{ke}\epsilon_{jkcd}\epsilon_{ikab}[jd|ce][ib|ae]\Big([jd|kc][ib|ka]-2[jd|kc][ia|kb]+[jc|kd][ia|kb]\Big).
\end{aligned}
$$

3.3 タイミングデータ

表 1 は前節の計算例の計算時間をまとめたものである. 計測は CPU が Intel(R) Core(TM)2 Duo 2.20GHz, メモリが 1GB, OS が WindowsXP(32bit) の計算機を使って行った. この結果から, QC²AS に実装した簡約アルゴリズムは実用に足りる性能を持っていると言える.

入力	計算時間 [ms]	簡約前の項数	簡約後の項数
`SimplifyMP(MP2())`	16 ms	3	2
`SimplifyMP(MP3())`	168 ms	48	10
`SimplifyMP(MP4S())`	10,547 ms	128	14

表 1: QC²AS による代数表式の簡約にかかった時間

4 まとめと今後の課題

[18] で着想を得, 実現に向けて開発を行ってきた数式処理システム QC²AS は, その骨子ができあがり, 数式処理システムとしての第一歩を踏み出した. 本稿ではその開発の報告を行い, この実装が実用レベルの問題である $E_{\mathrm{MP4}(S)}$ の簡約を行うのに十分な性能を持っていることを示した. また, QC²AS の設計の骨組みについて紹介した. QC²AS のシステム構成は単純で, 各クラスは高い独立性を保っている. このことが, システムの保守や拡張を容易にしている.

今後の課題として, 量子化学の記号計算を紹介する際に述べた, スピン積分の実装が挙げられる. スピン積分をしてようやく代数表式を数値計算に使う準備ができる. ここから数値計算のためのコードを生成するなどの展開がある. また, 代数表式を導出する理論 [17] をシステム上に組み込み, 代数表式の自動導出にも着手したい.

QC²AS は非オープンソースのフリーウェアとして配布されており, 最新版のバイナリは QC²AS のウェブサイト (`http://www2.rikkyo.ac.jp/web/fullmoon/qc2as/`) で入手できる.

謝辞

本研究は立教大学学術推進特別重点資金の支援のもと行われている.

参 考 文 献

[1] W. Adams and P. Loustaunau. *An Introduction to Gröbner Bases*, Vol. 3 of *Graduate Studies in Mathematics*. AMS, 1994.

[2] A. Auer, G. Baumgartner, and D. E. Bernholdt. Automated code generation for many-body electronic structure methods: the tensor contraction engine. *Molecular Physics*, Vol. 104, pp. 211–228, 2006.

[3] R. J. Bartlett. Many-body perturbation theory and coupled-cluster theory for electron correlation in molecules. *Annual Review of Physical Chemistry*, Vol. 32, pp. 359–401, 1981.

[4] R. J. Bartlett and M. Musial. Coupled-cluster theory in quantum chemistry. *Reviews of Modern Physics*, Vol. 79, pp. 291–352, 2007.

[5] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer-Verlag, New York, second edition, 1996.

[6] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer-Verlag, New York, 1998.

[7] D. G. Fedorov and K. Kitaura, editors. *The fragment molecular orbital method: practical applications to large molecular systems*. CRC Press, New York, 2009.

[8] J. B. Foresman and Æ. Frisch. *Exploring Chemistry with Electronic Structure Methods*. Gaussian, Inc., second edition, 1996.

[9] Gordon Group/GAMESS Homepage. http://www.msg.ameslab.gov/gamess/.

[10] S. Hirata. Tensor contraction engine: abstraction and automated parallel implementation of configuration-interaction, coupled-cluster, and many-body perturbation theories. *Journal of Physical Chemistry A*, Vol. 107, pp. 9887–9897, 2003.

[11] S. Hirata. Symbolic algebra in quantum chemistry. *Theoretical Chemistry Accounts*, Vol. 116, pp. 2–17, 2007.

[12] M. Minimair and P. Barnett. Solving polynomial equations for chemical problems using gröbner bases. *Molecular Physics*, Vol. 102, pp. 2521–2535, 2004.

[13] T. Nakano, T. Kaminuma, T. Sato, K. Fukuzawa, Y. Akiyama, M. Uebayasi, and K. Kitaura. Fragment molecular orbital method: use of approximate electrostatic potential. *Chemical Physics Letters*, Vol. 351, pp. 475–480, 2002.

[14] T. Osoekawa, N. Shinohara, Y. Mochizuki, and K. Yokoyama. Gröbner basis technique for algebraic formulas in electron correlation theories. *Proceeding of the 10th International Conference on Computational Science and Its Applications (ICCSA-2010)*, pp. 17–23, 2010.

[15] RCSB Protein Data Bank. http://www.pdb.orb/.

[16] I. Shavitt and R. J. Bartlett. *Many-Body Methods in Chemistry and Physics*. Cambridge University Press, London, 2009.

[17] A. Szabo and N. S. Ostlund. *Modern Quantum Chemistry*. MacMillan, New York, 1982.

[18] 小副川健, 望月祐志, 横山和弘. 電子相関理論のための数式処理システムに向けて. 京都大学数理解析研究所講究録, 巻号未定, , 2010.